**Problem No:01**

**Problem Statement: UDP (Clint-server) implementation using c programming in Linux terminal.**

**Source Code:**

**Server code:**

// Server side implementation of UDP client-server model

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#define PORT 8080

#define MAXLINE 1024

// Driver code

int main() {

int sockfd;

char buffer[MAXLINE];

char \*hello = "Hello from server";

struct sockaddr\_in servaddr, cliaddr;

// Creating socket file descriptor

if ( (sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0 ) {

perror("socket creation failed");

exit(EXIT\_FAILURE);

}

memset(&servaddr, 0, sizeof(servaddr));

memset(&cliaddr, 0, sizeof(cliaddr));

// Filling server information

servaddr.sin\_family = AF\_INET; // IPv4

servaddr.sin\_addr.s\_addr = INADDR\_ANY;

servaddr.sin\_port = htons(PORT);

// Bind the socket with the server address

if ( bind(sockfd, (const struct sockaddr \*)&servaddr,

sizeof(servaddr)) < 0 )

{

perror("bind failed");

exit(EXIT\_FAILURE);

}

int len, n;

len = sizeof(cliaddr); //len is value/resuslt

n = recvfrom(sockfd, (char \*)buffer, MAXLINE,

MSG\_WAITALL, ( struct sockaddr \*) &cliaddr,

&len);

buffer[n] = '\0';

printf("Client : %s\n", buffer);

sendto(sockfd, (const char \*)hello, strlen(hello),

MSG\_CONFIRM, (const struct sockaddr \*) &cliaddr,

len);

printf("Hello message sent.\n");

return 0;

}

**Client Code:**

// Client side implementation of UDP client-server model

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#define PORT 8080

#define MAXLINE 1024

// Driver code

int main() {

int sockfd;

char buffer[MAXLINE];

char \*hello = "Hello from client";

struct sockaddr\_in servaddr;

// Creating socket file descriptor

if ( (sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0 ) {

perror("socket creation failed");

exit(EXIT\_FAILURE);

}

memset(&servaddr, 0, sizeof(servaddr));

// Filling server information

servaddr.sin\_family = AF\_INET;

servaddr.sin\_port = htons(PORT);

servaddr.sin\_addr.s\_addr = INADDR\_ANY;

int n, len;

sendto(sockfd, (const char \*)hello, strlen(hello),

MSG\_CONFIRM, (const struct sockaddr \*) &servaddr,

sizeof(servaddr));

printf("Hello message sent.\n");

n = recvfrom(sockfd, (char \*)buffer, MAXLINE,

MSG\_WAITALL, (struct sockaddr \*) &servaddr,

&len);

buffer[n] = '\0';

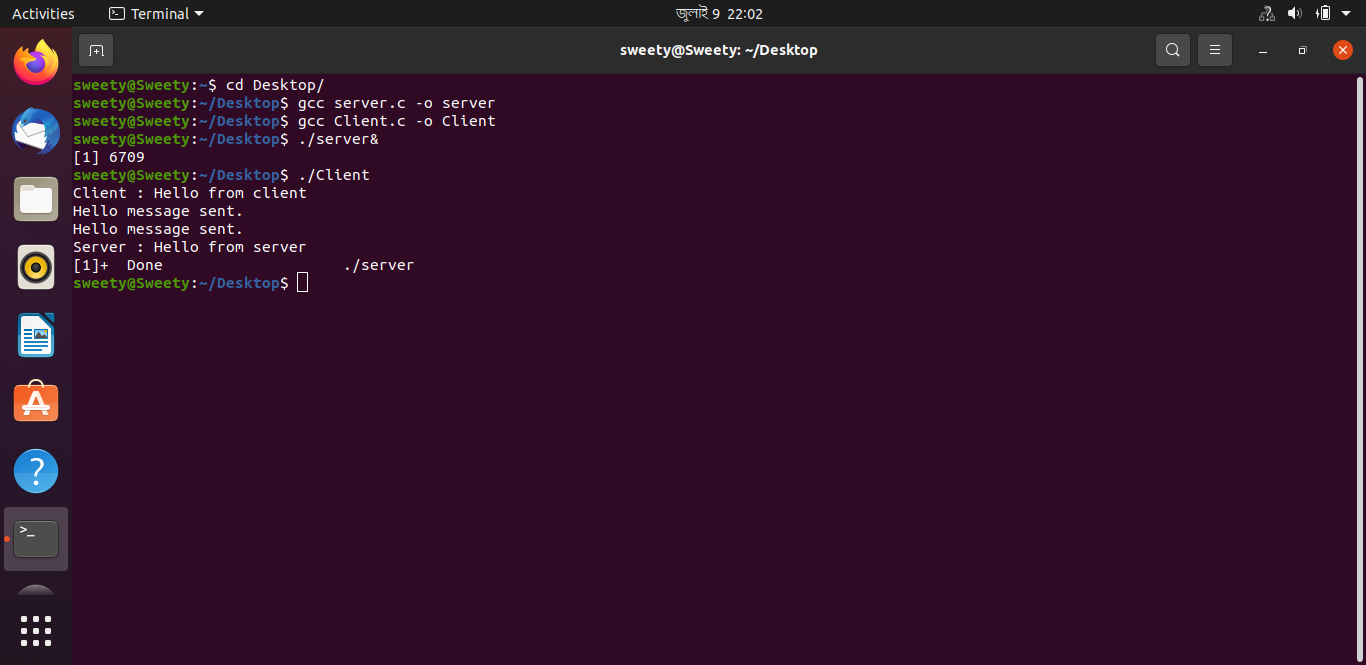
printf("Server : %s\n", buffer);

close(sockfd);

return 0;

}

**Sample Input Output:**



**Problem No:02**

**Problem Statement: UDP file transfer between server and Client using c programming in Linux terminal.**

**Source Code:**

**Server code:**

// server code for UDP socket programming

#include <arpa/inet.h>

#include <netinet/in.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <unistd.h>

#define IP\_PROTOCOL 0

#define PORT\_NO 15050

#define NET\_BUF\_SIZE 32

#define cipherKey 'S'

#define sendrecvflag 0

#define nofile "File Not Found!"

// function to clear buffer

void clearBuf(char\* b)

{

int i;

for (i = 0; i < NET\_BUF\_SIZE; i++)

b[i] = '\0';

}

// function to encrypt

char Cipher(char ch)

{

return ch ^ cipherKey;

}

// function sending file

int sendFile(FILE\* fp, char\* buf, int s)

{

int i, len;

if (fp == NULL) {

strcpy(buf, nofile);

len = strlen(nofile);

buf[len] = EOF;

for (i = 0; i <= len; i++)

buf[i] = Cipher(buf[i]);

return 1;

}

char ch, ch2;

for (i = 0; i < s; i++) {

ch = fgetc(fp);

ch2 = Cipher(ch);

buf[i] = ch2;

if (ch == EOF)

return 1;

}

return 0;

}

// driver code

int main()

{

int sockfd, nBytes;

struct sockaddr\_in addr\_con;

int addrlen = sizeof(addr\_con);

addr\_con.sin\_family = AF\_INET;

addr\_con.sin\_port = htons(PORT\_NO);

addr\_con.sin\_addr.s\_addr = INADDR\_ANY;

char net\_buf[NET\_BUF\_SIZE];

FILE\* fp;

// socket()

sockfd = socket(AF\_INET, SOCK\_DGRAM, IP\_PROTOCOL);

if (sockfd < 0)

printf("\nfile descriptor not received!!\n");

else

printf("\nfile descriptor %d received\n", sockfd);

// bind()

if (bind(sockfd, (struct sockaddr\*)&addr\_con, sizeof(addr\_con)) == 0)

printf("\nSuccessfully binded!\n");

else

printf("\nBinding Failed!\n");

while (1) {

printf("\nWaiting for file name...\n");

// receive file name

clearBuf(net\_buf);

nBytes = recvfrom(sockfd, net\_buf,

NET\_BUF\_SIZE, sendrecvflag,

(struct sockaddr\*)&addr\_con, &addrlen);

fp = fopen(net\_buf, "r");

printf("\nFile Name Received: %s\n", net\_buf);

if (fp == NULL)

printf("\nFile open failed!\n");

else

printf("\nFile Successfully opened!\n");

while (1) {

// process

if (sendFile(fp, net\_buf, NET\_BUF\_SIZE)) {

sendto(sockfd, net\_buf, NET\_BUF\_SIZE,

sendrecvflag,

(struct sockaddr\*)&addr\_con, addrlen);

break;

}

// send

sendto(sockfd, net\_buf, NET\_BUF\_SIZE,

sendrecvflag,

(struct sockaddr\*)&addr\_con, addrlen);

clearBuf(net\_buf);

}

if (fp != NULL)

fclose(fp);

}

return 0;

}

**Client Code:**

// client code for UDP socket programming

#include <arpa/inet.h>

#include <netinet/in.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <unistd.h>

#define IP\_PROTOCOL 0

#define IP\_ADDRESS "127.0.0.1" // localhost

#define PORT\_NO 15050

#define NET\_BUF\_SIZE 32

#define cipherKey 'S'

#define sendrecvflag 0

// function to clear buffer

void clearBuf(char\* b)

{

int i;

for (i = 0; i < NET\_BUF\_SIZE; i++)

b[i] = '\0';

}

// function for decryption

char Cipher(char ch)

{

return ch ^ cipherKey;

}

// function to receive file

int recvFile(char\* buf, int s)

{

int i;

char ch;

for (i = 0; i < s; i++) {

ch = buf[i];

ch = Cipher(ch);

if (ch == EOF)

return 1;

else

printf("%c", ch);

}

return 0;

}

// driver code

int main()

{

int sockfd, nBytes;

struct sockaddr\_in addr\_con;

int addrlen = sizeof(addr\_con);

addr\_con.sin\_family = AF\_INET;

addr\_con.sin\_port = htons(PORT\_NO);

addr\_con.sin\_addr.s\_addr = inet\_addr(IP\_ADDRESS);

char net\_buf[NET\_BUF\_SIZE];

FILE\* fp;

// socket()

sockfd = socket(AF\_INET, SOCK\_DGRAM,

IP\_PROTOCOL);

if (sockfd < 0)

printf("\nfile descriptor not received!!\n");

else

printf("\nfile descriptor %d received\n", sockfd);

while (1) {

printf("\nPlease enter file name to receive:\n");

scanf("%s", net\_buf);

sendto(sockfd, net\_buf, NET\_BUF\_SIZE,

sendrecvflag, (struct sockaddr\*)&addr\_con,

addrlen);

printf("\n---------Data Received---------\n");

while (1) {

// receive

clearBuf(net\_buf);

nBytes = recvfrom(sockfd, net\_buf, NET\_BUF\_SIZE,

sendrecvflag, (struct sockaddr\*)&addr\_con,

&addrlen);

// process

if (recvFile(net\_buf, NET\_BUF\_SIZE)) {

break;

}

}

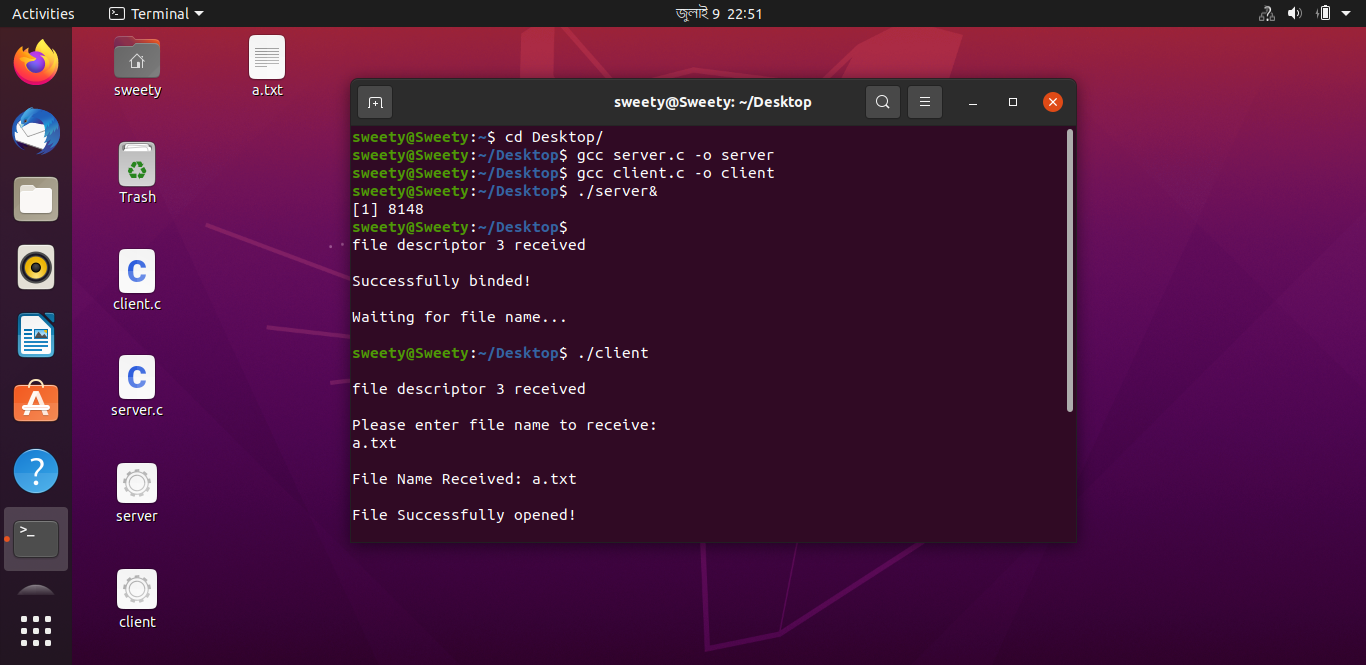
printf("\n-------------------------------\n");

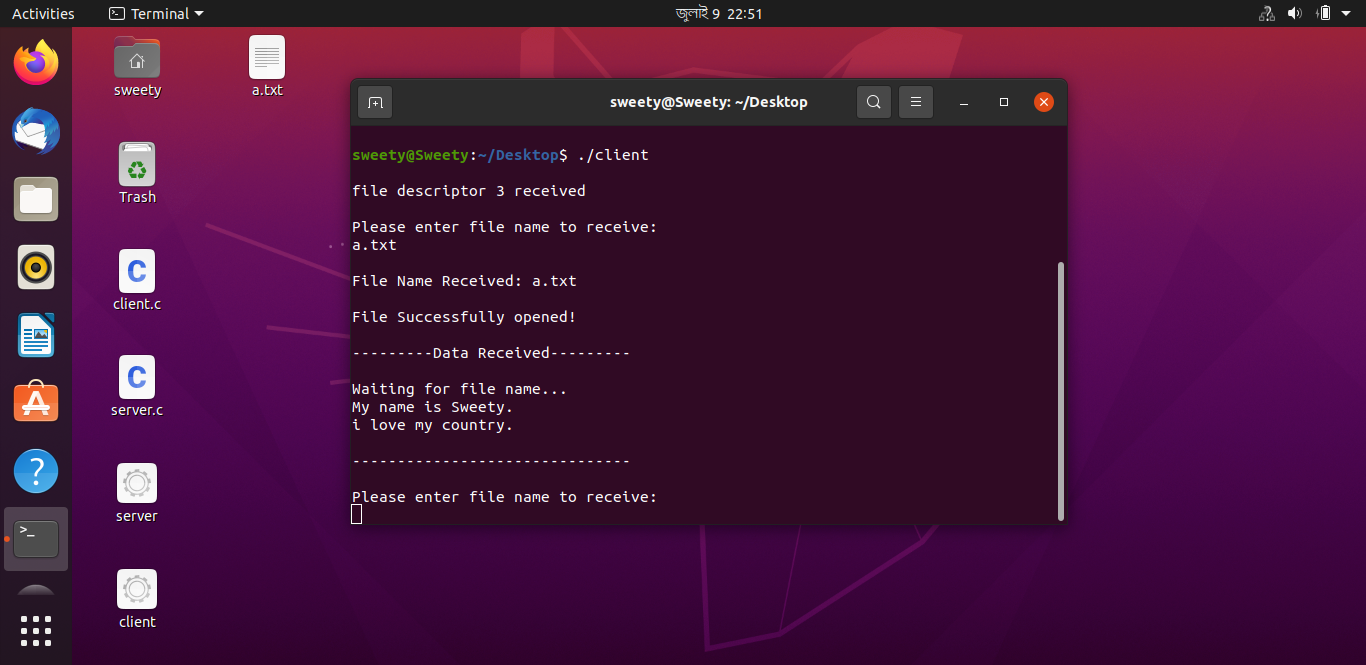
}

return 0;

}

**Sample Input Output:**





**Problem No:03**

**Problem Statement: Concurrent echo Clint-server interaction using c programming in Linux terminal.**

**Source Code:**

**Server code:**

#include<stdio.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<stdlib.h>

#include<arpa/inet.h>

#include<unistd.h>

void str\_echo(int s)

{

char buf[50];

//receiving data from client

recv(s,buf,50,0);

puts("Message from Client...");

fputs(buf,stdout);

send(s,buf,50,0);

}

int main()

{

int ls,cs,len;

struct sockaddr\_in serv,cli;

pid\_t pid;

puts("I am Server...");

//creating socket

ls=socket(AF\_INET,SOCK\_STREAM,0);

puts("Socket Created Successfully...");

//socket address structure

serv.sin\_family=AF\_INET;

serv.sin\_addr.s\_addr=INADDR\_ANY;

serv.sin\_port=htons(5000);

bind(ls,(struct sockaddr\*)&serv,sizeof(serv));

puts("Binding Done...");

listen(ls,3);

puts("Listening for Client...");

for(; ;)

{

len=sizeof(cli);

//accepting client connection

cs=accept(ls,(struct sockaddr\*)&cli,&len);

puts("\nConnected to Client...");

//creating child process

if((pid=fork()) == 0)

{

puts("Child process created...");

close(ls);

str\_echo(cs);

close(cs);

exit(0);

}

close(cs);

}

return 0;

}

**Client code:**

#include<stdio.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<unistd.h>

#include<stdlib.h>

#define MAX 100

void str\_echo(int s)

{

char buf[MAX],buf1[MAX];

puts("Enter the Message...");

fgets(buf,MAX,stdin);

printf(" mesg= %s\n",buf);

//fgets(buf);

//send(s,buf,MAX,0); //sending data to server

sendto(s,buf,MAX,0,NULL,0);

//receiving data from server

//recv(s,buf1,MAX,0);

recvfrom(s,buf1,MAX,0,NULL,NULL);

puts("Message from Server...");

fputs(buf1,stdout);

printf("\n mesg1 = %s\n",buf1);

//puts(buf1);

}

int main()

{

int ls;

struct sockaddr\_in cli;

puts("I am Client...");

/\*creating socket\*/

ls=socket(AF\_INET,SOCK\_STREAM,0);

puts("Socket Created Successfully...");

/\*socket address structure\*/

cli.sin\_family=AF\_INET;

// cli.sin\_addr.s\_addr=s6\_addr("127.0.0.1");

cli.sin\_addr.s\_addr=inet\_addr("127.0.0.1");

cli.sin\_port=htons(5000);

/\*connecting to server\*/

connect(ls,(struct sockaddr\*)&cli,sizeof(cli));

puts("Connected with Server...");

str\_echo(ls);

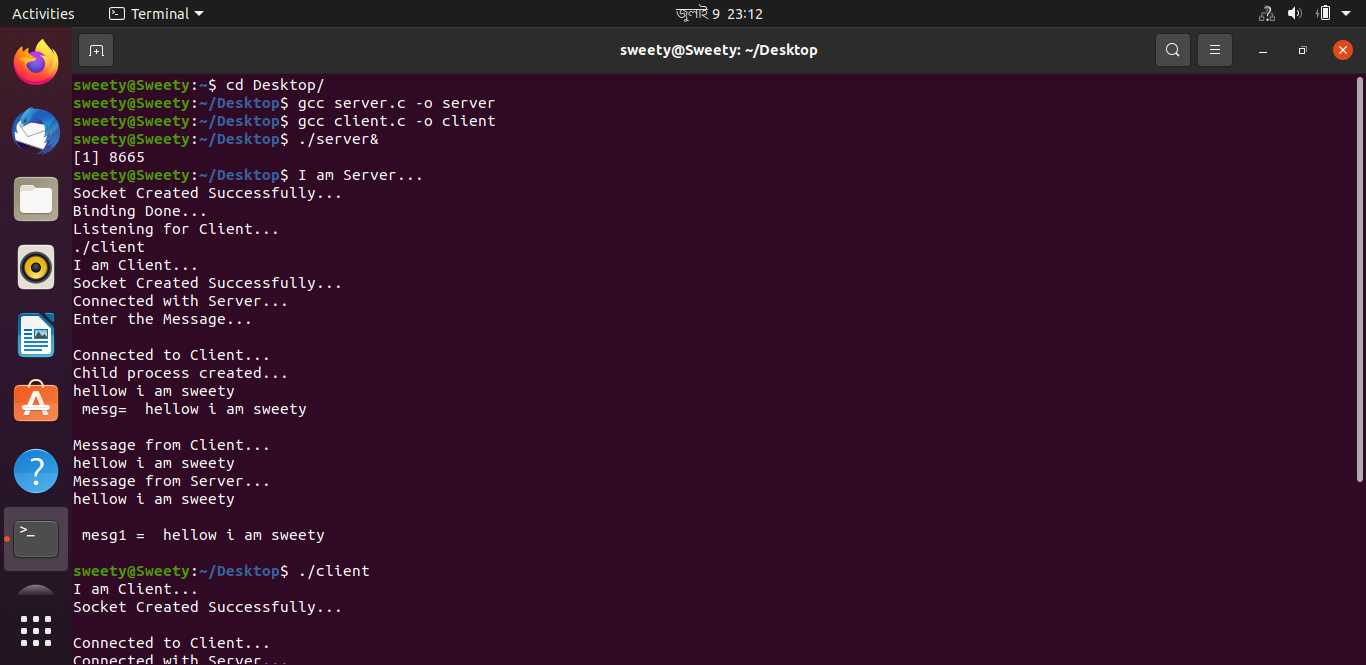
//pclose( ls);

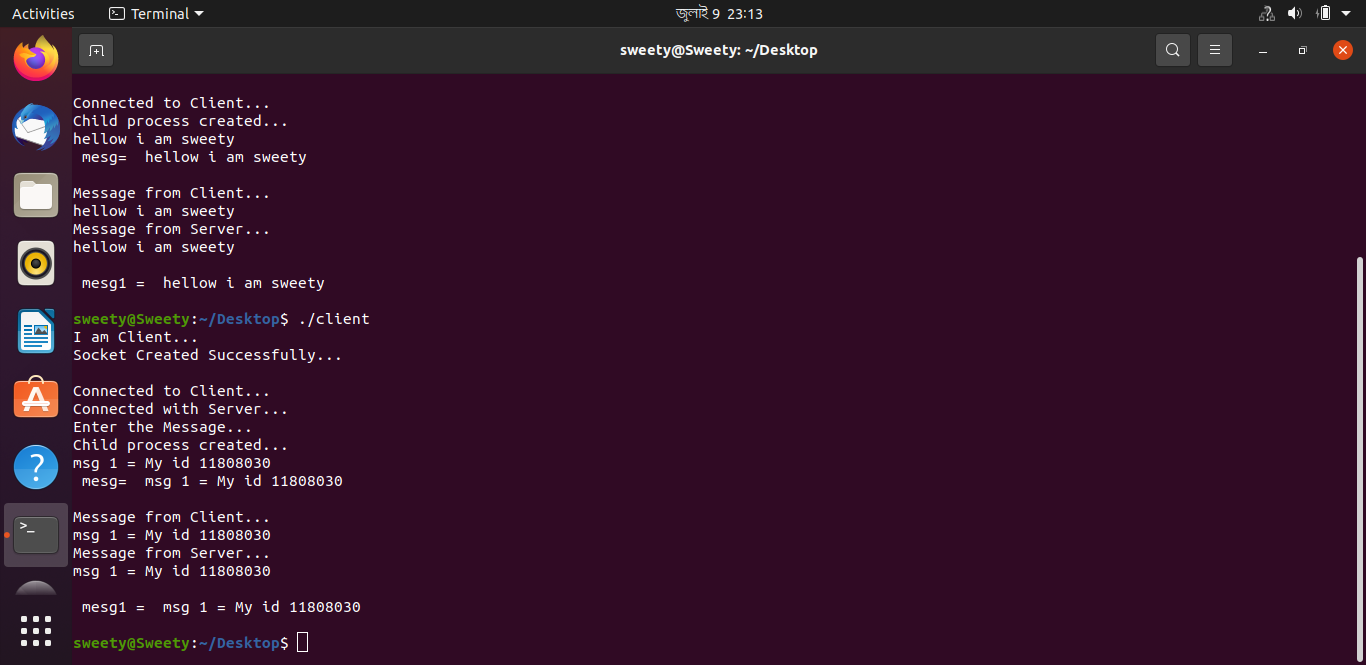
close(ls);

//close(2);

return 0;

}

**Sample Input Output:**



**Problem No:04**

**Problem Statement: Iterative Client-server using c programming in Linux terminal.**

**Source Code:**

**Server code:**

#include <stdlib.h>

#include <stdio.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <unistd.h>

#define MAXLINE 4096 /\*max text line length\*/

#define SERV\_PORT 3000 /\*port\*/

#define LISTENQ 8 /\*maximum number of client connections \*/

int main (int argc, char \*\*argv)

{

int listenfd, connfd, n;

socklen\_t clilen;

char buf[MAXLINE];

struct sockaddr\_in cliaddr, servaddr;

//creation of the socket

listenfd = socket (AF\_INET, SOCK\_STREAM, 0);

//preparation of the socket address

servaddr.sin\_family = AF\_INET;

servaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

servaddr.sin\_port = htons(SERV\_PORT);

bind (listenfd, (struct sockaddr \*) &servaddr, sizeof(servaddr));

listen (listenfd, LISTENQ);

printf("%s\n","Server running...waiting for connections.");

for ( ; ; ) {

clilen = sizeof(cliaddr);

connfd = accept (listenfd, (struct sockaddr \*) &cliaddr, &clilen);

printf("%s\n","Received request...");

while ( (n = recv(connfd, buf, MAXLINE,0)) > 0) {

printf("%s","String received from and resent to the client:");

puts(buf);

send(connfd, buf, n, 0);

}

if (n < 0) {

perror("Read error");

exit(1);

}

close(connfd);

}

//close listening socket

close (listenfd);

}

**Client code:**

#include<stdio.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<unistd.h>

#include<stdlib.h>

#define MAX 100

void str\_echo(int s)

{

char buf[MAX],buf1[MAX];

puts("Enter the Message...");

fgets(buf,MAX,stdin);

printf(" mesg= %s\n",buf);

//fgets(buf);

//send(s,buf,MAX,0); //sending data to server

sendto(s,buf,MAX,0,NULL,0);

//receiving data from server

//recv(s,buf1,MAX,0);

recvfrom(s,buf1,MAX,0,NULL,NULL);

puts("Message from Server...");

fputs(buf1,stdout);

printf("\n mesg1 = %s\n",buf1);

//puts(buf1);

}

int main()

{

int ls;

struct sockaddr\_in cli;

puts("I am Client...");

/\*creating socket\*/

ls=socket(AF\_INET,SOCK\_STREAM,0);

puts("Socket Created Successfully...");

/\*socket address structure\*/

cli.sin\_family=AF\_INET;

// cli.sin\_addr.s\_addr=s6\_addr("127.0.0.1");

cli.sin\_addr.s\_addr=inet\_addr("127.0.0.1");

cli.sin\_port=htons(5000);

/\*connecting to server\*/

connect(ls,(struct sockaddr\*)&cli,sizeof(cli));

puts("Connected with Server...");

str\_echo(ls);

//pclose( ls);

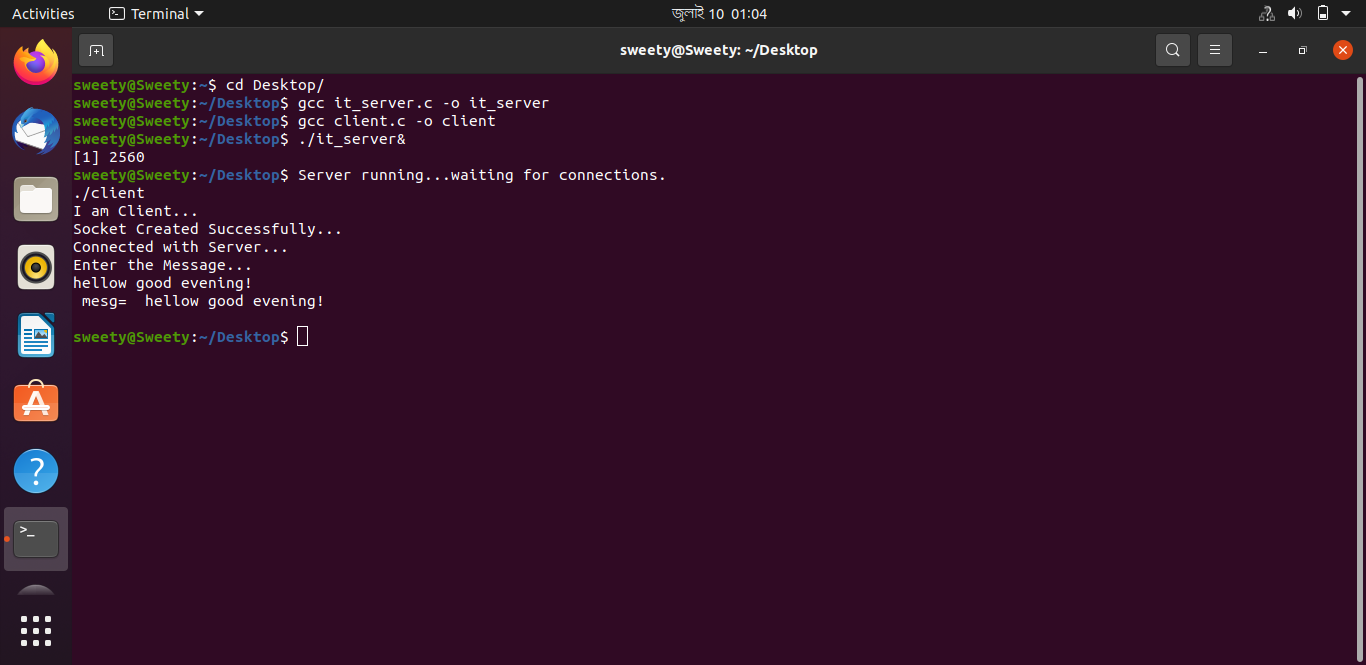
close(ls);

//close(2);

return 0;

}

**Sample Input Output:**



**Problem No:05**

**Problem Statement: RMI (client-server) test (Addition of two numbers) using java.**

**Source Code:**

**Server code:**

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.rmi.server.UnicastRemoteObject;

public class server extends UnicastRemoteObject implements adder {

public server() throws RemoteException{

//super();

}

@Override

public int add(int n1, int n2) throws RemoteException {

return n1+n2;

}

public static void main(String[] args) throws RemoteException{

try{

Registry reg=LocateRegistry.createRegistry(9999);

reg.rebind("hi server", new server());

System.out.println("server is ready");

}

catch(RemoteException e){

System.out.println("exception"+ e);

}

}

}

**Client code:**

import java.rmi.NotBoundException;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.util.Scanner;

public class client {

public static void main(String[] args) throws RemoteException, NotBoundException {

client c=new client();

c.connectRemote();

}

private void connectRemote() throws RemoteException, NotBoundException {

try{

Scanner sc=new Scanner(System.in);

Registry reg = LocateRegistry.getRegistry("localhost",9999);

adder ad=(adder)reg.lookup("hi server");

System.out.println("Enter Two Number");

int a=sc.nextInt();

int b=sc.nextInt();

System.out.println("Addition is "+ad.add(a,b));

}

catch(RemoteException e){

System.out.println("exception"+ e);

}

}

}

**Adder:**

import java.rmi.\*;

public interface adder extends Remote {

public int add(int n1, int n2) throws RemoteException;

}

**Sample Input Output:**

|  |  |
| --- | --- |
|  |  |

**Problem No:06**

**Problem Statement: ChatBot using client-server interaction using java.**

**Source Code:**

**Server code:**

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintStream;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.Scanner;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

\*

\* @author Chayti

\*/

public class Server {

int port;

ServerSocket server=null;

Socket client=null;

ExecutorService pool = null;

int clientcount=0;

public static void main(String[] args) throws IOException {

Server serverobj=new Server(5000);

serverobj.startServer();

}

Server(int port){

this.port=port;

pool = Executors.newFixedThreadPool(5);

}

public void startServer() throws IOException {

server=new ServerSocket(5000);

System.out.println("Server Booted");

System.out.println("Any client can stop the server by sending -1");

while(true)

{

client=server.accept();

clientcount++;

ServerThread runnable= new ServerThread(client,clientcount,this);

pool.execute(runnable)

private static class ServerThread implements Runnable {

Server server=null;

Socket client=null;

BufferedReader cin;

PrintStream cout;

Scanner sc=new Scanner(System.in);

int id;

String s;

ServerThread(Socket client, int count ,Server server ) throws IOException {

this.client=client;

this.server=server;

this.id=count;

System.out.println("Connection "+id+"established with client "+client);

cin=new BufferedReader(new InputStreamReader(client.getInputStream()));

cout=new PrintStream(client.getOutputStream());

}

@Override

public void run() {

int x=1;

try{

while(true){

s=cin.readLine();

System. out.print("Client("+id+") :"+s+"\n");

System.out.print("Server : ");

//s=stdin.readLine();

s=sc.nextLine();

if (s.equalsIgnoreCase("bye"))

{

cout.println("BYE");

x=0;

System.out.println("Connection ended by server");

break;

}

cout.println(s);

}

cin.close();

client.close();

cout.close();

if(x==0) {

System.out.println( "Server cleaning up." );

System.exit(0);

}

}

catch(IOException ex){

System.out.println("Error : "+ex);

}

}

}

}

**Client code:**

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.PrintStream;

import java.net.ServerSocket;

import java.net.Socket;

public class Client {

public static void main(String args[]) throws Exception {

Socket sk = new Socket("127.0.0.1", 5000);

BufferedReader sin = new BufferedReader(new InputStreamReader(sk.getInputStream()));

PrintStream sout = new PrintStream(sk.getOutputStream());

BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));

String s;

while (true) {

System.out.print("Client : ");

s = stdin.readLine();

sout.println(s);

if (s.equalsIgnoreCase("BYE")) {

System.out.println("Connection ended by client");

break;

}

s = sin.readLine();

System.out.print("Server : " + s + "\n");

}

sk.close();

sin.close();

sout.close();

stdin.close();

}

}

**Sample Input Output:**

|  |
| --- |
|  |
|  |

**Problem No:07**

**Problem Statement: Implementation Calculator(Interaction with client & server) using Java**

**Source Code:**

**Client Code:**

package calculator;

import java.net.MalformedURLException;

import java.rmi.Naming;

import java.rmi.NotBoundException;

import java.rmi.RemoteException;

import java.util.Scanner;

public class calculatorclient {

public static void main(String[] args) throws NotBoundException,RemoteException, MalformedURLException {

// TODO code application logic here

Scanner sc = new Scanner(System.in);

try

{

calculatorinterface c = (calculatorinterface) Naming.lookup("//localhost:1090/Calculator");

System.out.println("client is connected to server");

System.out.println("select any option from menu");

System.out.println("1.addition \n");

System.out.println("2.Substraction \n");

System.out.println("3.Multiplication \n");

System.out.println("4.Division \n");

int choice = sc.nextInt();

int x,y;

switch(choice)

{

case 1:

{

System.out.println("enter two number");

x = sc.nextInt();

y = sc.nextInt();

System.out.println("result:"+c.add(x, y));

}

case 2:

{

System.out.println("enter two number");

x = sc.nextInt();

y = sc.nextInt();

System.out.println("result:"+c.sub(x, y));

}

case 3:

{

System.out.println("enter two number");

x = sc.nextInt();

y = sc.nextInt();

System.out.println("result:"+c.mul(x, y));

}

case 4:

{

System.out.println("enter two number");

x = sc.nextInt();

y = sc.nextInt();

System.out.println("result:"+c.div(x, y));

}

}

}

catch(Exception e)

{

System.out.println(e);

}

}

}

**Server code:**

package calculator;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.rmi.NotBoundException;

public class calculatorserver {

public static void main(String[] args) throws RemoteException,NotBoundException

{

try

{

Registry r = java.rmi.registry.LocateRegistry.createRegistry(1090);

r.rebind("Calculator", new Calculator());

System.out.println("server is running");

}

catch(Exception e)

{

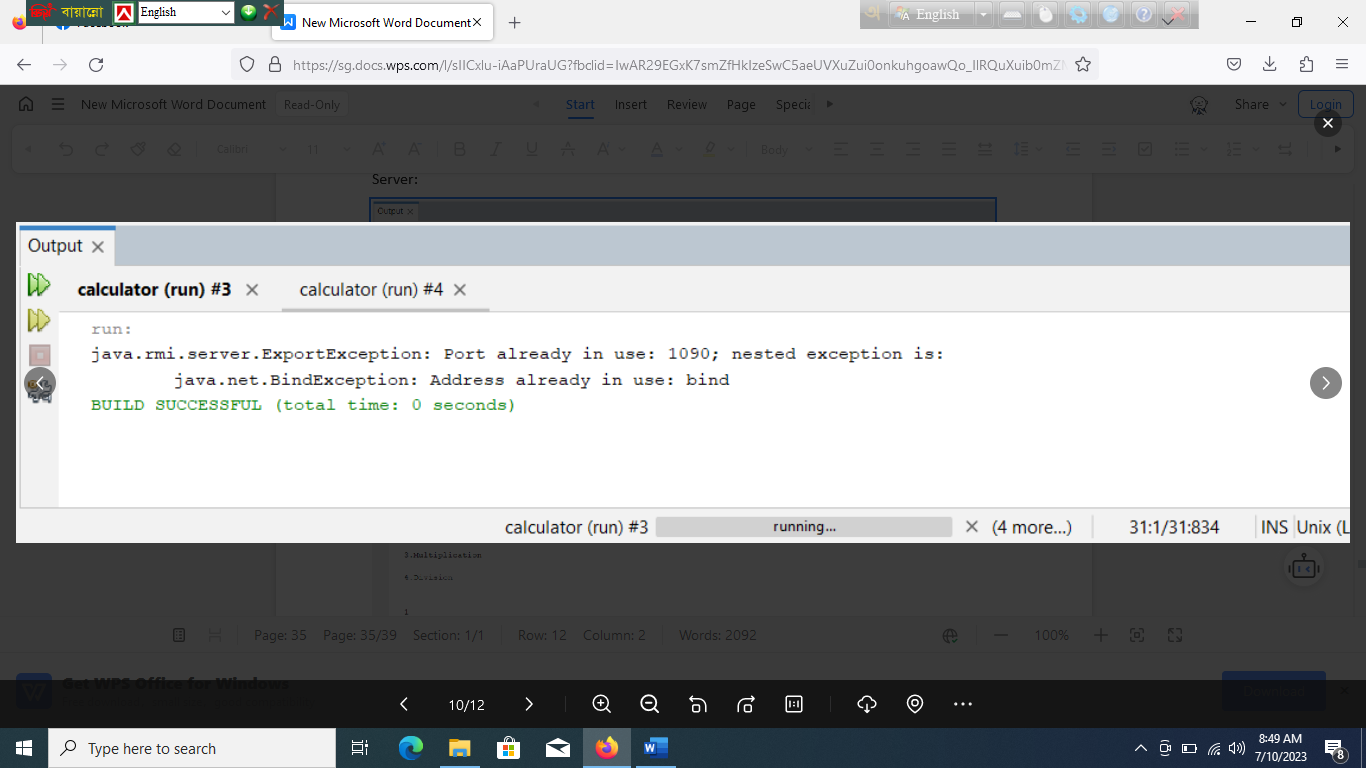
System.out.println(e);

} }

}

**Sample Input & Output:**

Server:



Client:

